UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/845,693	04/30/2001	Erik R. Altman	Y0R9-2000-0844 US (8728-4	2678

46069 7590 06/22/2005

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER	
HUISMAN, DAVID J	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 06/22/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/845,693	ALTMAN ET AL.	
	Examiner	Art Unit	
	David J. Huisman	2183	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *27 April 2005*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,5-13 and 15-21* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-13 and 15-21* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 April 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-3, 5-13, and 15-21 have been examined.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed of record in the file: Amendment as received on 4/27/2005.

Specification

3. The title of the invention is not descriptive. A new title is required that is clearly indicative of the invention to which the claims are directed. As will be described below, there is not necessarily a difference between instructions and compiler decoded instructions.

Claim Objections

4. Claim 1 is objected to because of the following informalities: Replace the phrase "wherein the first instruction form are decoded by a decoder in an execution pipeline and the second instruction form are predecoded by a compiler" with -- wherein the plurality of instructions of the first instruction form are decoded by a decoder in an execution pipeline and the plurality of instructions of the second instruction form are predecoded by a compiler--. The examiner requests that this change be made because it is not exactly clear how forms themselves are decoded/predecoded but it is clear how instructions of forms are decoded/predecoded. Appropriate correction is required.

Claim Rejections - 35 USC § 102

5. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

6. Claims 1-2, 5-9, 11, 13, and 16-19 are rejected under 35 U.S.C. 102(b) as being anticipated by Parady, U.S. patent No. 5,933,627.

7. Referring to claim 1, Parady has taught a method for processing a first instruction form (instructions from a first thread) and a second instruction form (instructions from at least a second thread) of an instruction set in a processor comprising the steps of:

a) providing a program of instructions comprising a plurality of instructions of the first instruction form and a plurality of instructions of the second instruction form, wherein the first instruction form are decoded by a decoder in an execution pipeline. See Fig.1 and Fig.3, component 14.

b) the second instruction form are predecoded by a compiler. Note from column 4, lines 24-25, that a compiler is disclosed. A compiler inherently decodes instructions written in a high-level language to instructions of a low-level language, which a processor is able to understand and execute. This decoding of the instructions is predecoding as it occurs prior to run-time.

c) storing the plurality of instructions of the second form in a plurality of buffers proximate to a plurality of execution units. See Fig.3 and note that instructions of the second form may belong to threads 1, 2, and 3, and are stored in buffers 104, 106, and 108, respectively.

d) executing at least one instruction of the first instruction form in response to a first counter. See Fig.3, component 110, and column 3, lines 50-56, and note that each thread has a counter for counting the address of the current instruction.

e) executing at least one instruction of the second instruction form in response to at least a second counter, wherein the second counter is invoked by a branch instruction of the first instruction form. Again, see Fig.3, component 110, and column 3, lines 50-56, and note that each thread has a counter for counting the address of the current instruction. Also, a second thread (having instructions of a second form) may be executed in response to a branch in the first thread. See column 4, line 65.

f) wherein the step of executing at least one instruction of the second instruction form further comprises the steps of:

f1) de-gating a plurality of execution queues storing the plurality of instructions of the first instruction form. Since only one thread executes until a thread switch occurs, then if instructions of a second form are executing (say from thread 3), then the rest of the execution queues, including the instruction queue holding instructions of the first form (say from thread 0) are de-gated (i.e., they do not provide instructions). See column 3, lines 37-43.

f2) pausing a fetching of the first instruction form from a memory. Clearly, by de-gating an execution queue, the queue will not be providing and instructions for execution. Therefore, fetching from the queue (memory) will be paused.

8. Referring to claim 2, Parady has taught a method as described in claim 1.

a) Parady has further taught that instructions of the first form and instructions of the second form are generated by a compiler. Although the generation is not explicitly disclosed, Parady does refer to a compilation step. See column 4, lines 24-25. Therefore, a compiler does exist and a compiler inherently generates instructions from higher-level instructions.

b) Parady has further taught that instructions of the second form are statically loaded into the plurality of buffers as control signals ready for execution. As defined by the American Heritage® Dictionary of the English Language, 3rd Edition, "statically" means "fixed" (see the attached definition). Clearly, each of the buffers is fixed to store a specific type of instruction. Assuming thread 2 and thread 3 instructions are instructions of the second form, buffer 108 in Fig.3 is fixed to store only thread 3 instructions whereas buffer 106 is fixed only to store thread 2 instructions. Therefore, the storing of the instructions of the second form is static. Furthermore, it should be realized that instructions are control signals as they signal the system to perform an operation.

9. Referring to claim 5, Parady has taught a method as described in claim 1. Parady has further taught that the step of executing at least one instruction of the second instruction form further comprises the steps of:

a) fetching at least one instruction of the second instruction form from a buffer of the plurality of buffers. See Fig.3 and assume that instructions from the thread 3 buffer 108 are of the second form. When it is thread 3's turn to execute, instructions will be fetched from buffer 108.

b) sequencing the at least one instruction of the second instruction form to the execution units. Note from Fig.3 that the instructions from the thread 3 buffer 108 would be dispatched to execution units 41 by dispatch unit 28.

10. Referring to claim 6, Parady has taught a method as described in claim 1. Furthermore, although not explicitly stated, the second instruction form of Parady may be a logical subset of the first instruction form. More specifically, each of the threads has the ability to include the same instructions (add, subtract, branch, load, etc.). So, if the first form of instructions (thread 1) includes an ADD, SUB, MULT, DIV, LOAD, and BRANCH, and the second form of instructions (thread 2, for instance) includes SUB, DIV, LOAD, and BRANCH, then the second form of instructions are a subset of the first form of instructions.

11. Referring to claim 7, Parady has taught a method as described in claim 1. Parady has further taught that the step of executing at least one instruction of the first instruction form further comprises the steps of:

a) fetching an instruction of the first form from a memory. This inherently occurs as instructions must be stored somewhere. Looking at Fig.1, the instructions would be fetched from cache 12.

b) decoding the instruction. See Fig.1 and Fig.3, component 14.

c) issuing the decoded instruction to at least one execution unit. See Fig.1 and Fig.3 and note that the instruction is issued (dispatched) to the execution units.

12. Referring to claim 8, Parady has taught a method as described in claim 1. Parady has further taught that a return to fetching of the first instruction form is signaled by a switch bit in a buffer of a branch unit storing instructions of the second form. Again, as the second form of instructions are executing, a branch to the first form of instructions may be encountered (column 4, line 65). If so, a switch will occur in response to at least a switch bit (signal from component 112 in Fig.3. All of the components involved in this switching process may be part of a "branch unit".

13. Referring to claim 9, Parady has taught a method as described in claim 1. Parady has further taught that a return to fetching of the first instruction form is signaled by a return instruction of the second instruction form stored in a buffer of a branch unit. See column 4, line 65. Note that a second thread (second form of instructions) may include a branch back to a first thread (first form of instructions), which may have branched to the second thread in the first place. Consequently, this branch in the second thread is a return instruction as control is passed back to the original thread.

14. Referring to claim 11, Parady has taught a processor for processing a program of instructions comprising instructions of a first instruction form (instructions from a first thread) and a second instruction form (instructions from another thread) comprising:

a) a plurality of execution units for receiving instructions. See Fig.1, components 32-46.

b) a branch unit (Fig.1, component 18) connected to an instruction fetch unit (Fig.1, component 16, for instance) for the first instruction form and a sequencer (Fig.3, component 112) for the second instruction form, wherein the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units. Looking at Fig.3, assume threads 0 and 1 include instructions of the first form while threads 2 and 3 include instructions of the second form. If thread 1 is executing, instructions are being fetching from the thread 1 buffer 104. If a thread switch occurs to thread 2, then thread switch logic will control the gates at each of the buffers to determine which buffer will provide instructions. This control line can be seen emanating from component 112 and going to each of the buffers. Inherently, there is some logic (gates) which takes the signal and uses it to make the buffer stop or start providing instructions.

c) a decode unit for decoding instructions of the first instruction form into control signals for the execution units. See Fig.3, component 14.

d) a plurality of buffers, proximate to the execution units, for storing predecoded instructions of the second instruction form. See Fig.3, components 106 and 108 and note that they hold instructions of the second form which are predecoded by predecoder 24 (Fig.1).

15. Referring to claim 13, Parady has taught a processor as described in claim 11. Parady has further taught that the sequencer, engaged by the branch unit, addresses the predecoded instructions of the second instruction form stored in the buffers and sequences predecoded instructions of the second instruction form to the execution units. In response to a branch (column 4, line 65) to a group of instructions of the second form (some thread), the second form of instructions will be addressed, i.e., fetched, and sequenced/dispatched to the execution units.

16. Referring to claim 16, Parady has taught a processor as described in claim 11. Parady has further taught that the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form. See column 4, line 65. Note that each thread may include this branch instruction.

17. Referring to claim 17, Parady has taught a processor as described in claim 11. Parady has further taught that the branch unit switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form. See column 4, line 65. Note that each thread may include this branch instruction.

18. Referring to claim 18, Parady has taught a processor as described in claim 11. Parady has further taught a switch bit in a buffer of the plurality of buffers connected to the branch unit signals the sequencer to stop fetching from the buffers and enables instruction fetching from a

memory storing instructions of the first instruction form. As the second form of instructions are executing, a branch to the first form of instructions may be encountered (column 4, line 65). If so, a switch will occur in response to at least a switch bit (signal from component 112 in Fig.3. All of the components involved in this switching process may be part of a "branch unit". This switch results in stopping the fetching from the buffer where the branch came from and enabling the buffer (memory) of the thread which is branched to (first form of instructions).

19. Referring to claim 19, Parady has taught a processor as described in claim 11. Parady has further taught that an execution bandwidth of the execution units is larger than a fetch/issue bandwidth of the first form. Looking at Fig.1, it can be seen that there are 8 execution units, so up to 8 instructions may execute concurrently (bandwidth of 8). However, component 30 shows that only 4 instructions are fetched/issued per cycle (bandwidth of 4). Therefore, the execution bandwidth is larger than the fetch/issue bandwidth.

Claim Rejections - 35 USC § 103

20. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

21. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above.

22. Referring to claim 3, Parady has taught a method as described in claim 2. Furthermore, although not explicitly stated, it is possible in Parady that instructions of the second form are

more frequently executed that instructions of the first form. For instance, the second form of instructions might include a branch which executes 1000 times while the first form of instructions might not include anything close to that magnitude. In this case, the second form of instructions would execute more frequently. There are many different situations in which the second form of instructions would execute more frequently than the first.

23. Claims 10, 15, and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above, in view of Hennessy and Patterson, "Computer Architecture - A Quantitative Approach, 2nd Edition," 1996 (herein referred to as Hennessy).

24. Referring to claim 10, Parady has taught a method as described in claim 1. Parady has not explicitly taught that each execution unit is associated with a different buffer of the plurality of buffers. However, Hennessy has taught that a buffer (reservation station) may exist for each functional unit so that execution of instructions for that unit is controlled. See the last paragraph on page 252 and also see Fig.4.8 on page 253. Reservation stations also allow for register renaming which has advantages that are well known in the art. Consequently, it would have been obvious to one of ordinary skill in the art to have a buffer correspond to each functional unit.

25. Referring to claim 15, Parady has taught a processor as described in claim 11. Furthermore, the processor of claim 15 performs the method of claim 10. Consequently, claim 15 is rejected for the same reasons set forth in the rejection of claim 10.

26. Referring to claim 21, Parady has taught a processor for processing a first instruction form (instructions from a first thread) and a second instruction form (instructions from at least a second thread) of an instruction set comprising:

a) a plurality of execution units for receiving instructions. See Fig.1, components 32-46.

b) a branch unit (Fig.1, component 18) connected to an instruction fetch unit (Fig.1, component 16, for instance) for the first instruction form and a sequencer (Fig.3, component 112) for the second instruction form, wherein the branch unit switches the processor from the first instruction form to the second instruction form in response to a branch instruction of the first instruction form and switches the processor from the second instruction form to the first instruction form in response to a branch instruction of the second instruction form. See column 4, line 65, and note that while executing instructions of the first or second form, a branch to a different thread may be executed.

c) a decode unit adapted to decode instructions of the first instruction form into control signals for the execution units. See Fig.1, component 14.

d) an issue unit adapted to sequence decoded instructions of the first instruction form. See Fig.1, component 28.

e) a plurality of buffers, proximate to the execution units, for statically storing predecoded instructions of the second instruction form. See Fig.3, components 106 and 108, for instance (assuming the second form includes thread 2 and thread 3 instructions). As defined by the American Heritage® Dictionary of the English Language, 3rd Edition, "statically" means "fixed" (see the attached definition). Clearly, each of the buffers is fixed to store a specific type of

instruction. Buffer 108 in Fig.3 is fixed to store only thread 3 instructions whereas buffer 106 is fixed only to store thread 2 instructions. Therefore, the storing is static.

f) Parady has not explicitly taught that each execution unit is connected to a corresponding buffer of the plurality of buffers. However, Hennessy has taught that a buffer (reservation station) may exist for each functional unit so that execution of instructions for that unit is controlled. See the last paragraph on page 252 and also see Fig.4.8 on page 253. Reservation stations also allow for register renaming which has advantages that are well known in the art. Consequently, it would have been obvious to one of ordinary skill in the art to have a buffer correspond to each functional unit.

g) the sequencer, engaged by the branch unit, adapted to fetch the predecoded instructions and sequence the predecoded instructions of the second instruction form, wherein the sequencer is connected to a plurality of gates connected between a plurality of execution queues adapted to store the decoded instructions of the first instruction form and the plurality of execution units, the sequencer further adapted to control the gates. Looking at Fig.3, assume threads 0 and 1 include instructions of the first form while threads 2 and 3 include instructions of the second form. If thread 1 is executing, instructions are being fetching from the thread 1 buffer 104. If a thread switch occurs to thread 2, then thread switch logic will control each of the buffers to determine which buffer will provide instructions. This control line can be seen emanating from component 112 and going to each of the buffers. Inherently, there is some logic (gates) which takes the signal and uses it to make the buffer stop or start providing instructions.

27. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above, in view of Ball and Larus, "Efficient Path Profiling," 1996 (as previously cited and herein referred to as Ball).

28. Referring to claim 12, Parady has taught a processor as described in claim 11.

a) Parady has not taught that the instructions of the first form and instructions of the second form are generated based on execution frequency. However, Ball has taught by measuring execution frequency, profile-driven compilation (generation) may be achieved so that the generation is optimized. See the first paragraph of the introduction on page 46, column 1. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady to include generation based on frequency.

b) Furthermore, although not explicitly stated, it is possible in Parady that instructions of the second form are more frequently executed that instructions of the first form. For instance, the second form of instructions might include a branch which executes 1000 times while the first form of instructions might not include anything close to that magnitude. In this case, the second form of instructions would execute more frequently. There are many different situations in which the second form of instructions would execute more frequently than the first.

29. Claim 12 is rejected under 35 U.S.C. 103(a) as being unpatentable over Parady, as applied above, in view of Lavi et al., U.S. Patent No. 6,453,407 (as previously applied and herein referred to as Lavi).

30. Referring to claim 20, Parady has taught a processor as described in claim 11.

a) Furthermore, although not explicitly stated, the second instruction form of Parady may be a logical subset of the first instruction form. More specifically, each of the threads has the ability to include the same instructions (add, subtract, branch, load, etc.). So, if the first form of instructions (thread 1) includes an ADD, SUB, MULT, DIV, LOAD, and BRANCH, and the second form of instructions (thread 2, for instance) includes SUB, DIV, LOAD, and BRANCH, then the second form of instructions are a subset of the first form of instructions.

b) the predecoded instructions of the second instruction form are statically stored in the plurality of buffers. See Fig.3, components 106 and 108, for instance (assuming the second form includes thread 2 and thread 3 instructions). As defined by the American Heritage® Dictionary of the English Language, 3rd Edition, "statically" means "fixed" (see the attached definition). Clearly, each of the buffers is fixed to store a specific type of instruction. Buffer 108 in Fig.3 is fixed to store only thread 3 instructions whereas buffer 106 is fixed only to store thread 2 instructions. Therefore, the storing is static. Note also, the instructions are predecoded (Fig.1, component 24).

c) Parady has not taught that the predecoded instructions of the second instruction form are control signals generated by a compiler and are not decoded during a runtime of the program. However, Lavi has taught such a concept. See column 11, lines 25-59, and note that instructions may be predecoded by the compiler and stored in a buffer so that runtime decoding for those instructions may be avoided. This results in reduced hardware complexity and power consumption. As a result, it would have been obvious to one of ordinary skill in the art at the time of the invention to modify Parady such that instructions are predecoded by the compiler and not decoded during run-time.

Response to Arguments

31. Applicant's arguments filed on April 27, 2005, have been fully considered but they are not persuasive.

32. Applicant argues the novelty/rejection of claim 1 on page 10 of the remarks, in substance that:

> "Parady teaches an instruction cache and a decode unit through which all instructions are processed (see Figure 1 and col. 2, line 66-col. 3, line 10). The instructions of Parady are not provided predecoded by a compiler, essentially as claimed in claim 1. The instructions of Parady are processed by the decoder, and for off-chip instructions, processed by a predecoder and the decoder. Thus, the instructions of Parady are not predecoded as claimed in claim 1."

33. These arguments are not found persuasive for the following reasons:

a) As described in the rejection of claim 1 above, Parady has disclosed that a compiler exists. See column 4, lines 24-25. A compiler inherently decodes instructions written in a high-level language to instructions of a low-level language, which a processor is able to understand and execute. This decoding of the instructions is predecoding as it occurs prior to runtime. In addition, it should be realized that applicant has not claimed that these predecoded instructions are not decoded at runtime.

34. Applicant argues the novelty/rejection of claim 11 on pages 10-11 of the remarks, in substance that:

> "Parady has not taught that "...the sequencer controls a plurality of gates connected between a plurality of execution queues for storing decoded instructions of the first instruction form and the plurality of execution units" (emphasis added). The instruction buffers of Parady are controlled by thread switching logic (see element 112 of Figures 3). The thread switching logic is a pointer-based system for selecting a next thread from the instruction buffers (see col. 3, lines 57-65). The thread switching logic is not a gate connected between an execution queue for storing decoded instructions of the first instruction form and an execution unit, essentially as claimed in claim 11."

35. These arguments are not found persuasive for the following reasons:

a) As described in the rejection of claim 11 above, thread switching logic 112 determines which buffer is to provide instructions. While thread pointer 116 may assist in determining which thread is to execute next, logic 112 sends signals to the buffers so that the appropriate buffer may provide instructions. As is known, the building blocks of all hardware are logic gates. There must be logic gates that receive the signal from the thread switching logic 112, and based on that signal, cause a buffer to be active while causing others to be inactive. If there were no logic gates being controlled, then how is the activity and the inactivity of the buffers controlled and why are signals sent to the buffers from logic 112?

36. Regarding applicant's argument with respect to claim 2 that there is no loading of a control signal in Parady, the examiner asserts that instructions are control signals. Instructions are x-bit signals which control how the system operates. These signals are loaded into the buffers.

37. Applicant's argument of the claim 21 rejection is similar to the argument of the claim 11 rejection. Consequently, the examiner's response to the argument of the claim 11 rejection also applies to the argument of the claim 21 rejection.

Conclusion

38. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to David J. Huisman whose telephone number is (571) 272-4168. The examiner can normally be reached on Monday-Friday (8:00-4:30).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

DJH

David J. Huisman
June 15, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100